Interactive Level-of-Detail Selection Using Image-Based Quality Metric for Large Volume Visualization

Chaoli Wang, *Student Member, IEEE,* Antonio Garcia, and Han-Wei Shen

***Abstract*— For large volume visualization, an image-based quality metric is difficult to incorporate for level-of-detail selection and rendering without sacrificing the interactivity. This is because it is usually time-consuming to update view-dependent information as well as adjust to transfer function changes. In this paper, we introduce an image-based level-of-detail selection algorithm for interactive visualization of large volumetric data. The design of our quality metric is based on an efficient way to evaluate the contribution of multiresolution data blocks to the final image. To ensure real-time update of the quality metric and interactive level-of-detail decisions, we propose a summary table scheme in response to run-time transfer function changes, and a GPU-based solution for visibility estimation. Experimental results on large scientific and medical data sets demonstrate the effectiveness and efficiency of our algorithm.**

***Index Terms*— Data compaction and compression, perceptual reasoning, viewing algorithms, interaction techniques, hierarchical image representation, volume visualization.**

I. Introduction

DIRECT volume rendering with hardware texture mapping has become a standard technique for visualizing three-dimensional scalar fields from scientific and medical applications. An increasing number of these applications are now producing large-scale data sets, ranging from gigabytes to terabytes. One example is the Visible Woman (VisWoman) data set with resolution of $512 \times 512 \times 1728$ from The National Library of Medicine, generated as part of the Visible Human Project. Another example is the Richtmyer-Meshkov Instability (RMI) simulation performed at Lawrence Livermore National Laboratory. The simulation was executed on a $2048 \times 2048 \times 1920$ rectilinear grid, and it produced 7.5 gigabytes of data at each time step.

While it is common for the domain scientists to generate enormous amount of data, the size of video memory in the state-of-the-art high-end graphics hardware is limited to only several hundred megabytes. This disparity severely challenges brute-force conventional hardware-texturing based volume rendering approaches. New visualization systems that can scale adequately and ensure a high level of interactivity are needed. Among several alternatives, multiresolution volume rendering [13], [17], [31] is a solution that can reduce the rendering cost dramatically. To perform interactive rendering, a multiresolution data hierarchy composed of multiple spatially partitioned blocks is first created. At run time, as the user navigates through the hierarchy, various amounts of data at different levels of detail can be extracted and used for the rendering.

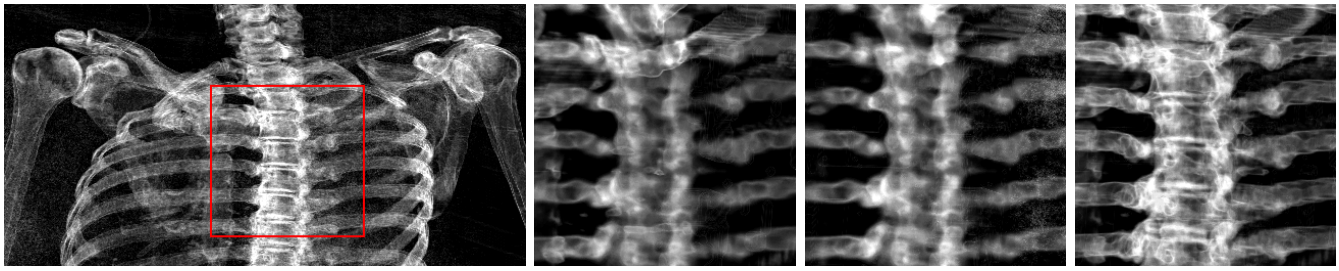
Often, such a level-of-detail (LOD) is determined by various user-specified parameters, such as the tolerance of errors based on certain data-dependent metrics [1], [19], [29], different view-dependent parameters [17], [21], or both [12], [13], [22], [32]. In general, these metrics can be classified as *data-based* and *image-based* metrics. Data-based metrics measure the distortion between low and high (or full) resolution data blocks in the volume. The most widely used data-based metrics are *mean square error* (MSE), L^2*-norm*, and *signal-to-noise ratio* (SNR). These metrics have clear physical meanings and are simple to compute. However, they are usually not effective in predicting the quality of the rendered images due to the lack of correlation between data and image, as indicated in [7], [15], [22], [27], [30]. Image-based metrics focus on the ultimate images the user perceives, and strive to capture the quality loss in the rendered images introduced by rendering low resolution data. These metrics are intrinsically view-dependent and more difficult to develop in conjunction with interactive LOD selections for large volume visualization. The major challenge lies in designing an image-based metric for quality-driven LOD selection, and updating the metric fast enough as not to harm the interactivity.

In this paper, we present an interactive LOD selection and rendering algorithm using an image-based quality metric for visualizing large volumetric data. The main contributions of our paper are:

- We introduce an image-space model for the quality metric design, based on an efficient way to evaluate the importance values of coarse-grained multiresolution data blocks on the final image. Fig. 1 shows a comparison of the LOD rendering of the VisWoman data set using our image-based quality metric and the MSE-based and SNRMSE-based (MSE of SNR) metrics. Unlike traditional LOD selection algorithms using data-based metrics, our LOD selection algorithm captures the structural distortion of the data and generates images of better visual quality under similar block budgets.
- Our method is adaptive to changes of the input transfer function. We utilize a zigzag run-length encoding scheme to store summary tables of data blocks in the mul-

C. Wang and H.-W. Shen are with the Department of Computer Science and Engineering, The Ohio State University, 395 Dreese Laboratories, 2015 Neil Avenue, Columbus, OH 43210. E-mail:{wangcha, hwshen}@cse.ohio-state.edu.

A. Garcia is with Intel Corporation, 2700 156th Avenue NE, Suite 300, Bellevue, WA 98007. E-mail:antonio.garcia@intel.com.



(a) (b) (c) (d)

Fig. 1. (a) shows a zoom to the upper skeleton of the VisWoman data set, rendered with full resolution. (b) (MSE-based, 36 blocks), (c) (SNRMSE-based, 37 blocks), and (d) (image-based, 34 blocks) show a closer zoom to the spine while rendered in low resolution.

tiresolution hierarchy with very small storage overhead (around 1% of the original volume data). During the run-time rendering, we can update the quality metric within seconds for large data sets whenever the transfer function changes at run time.

- Based on *summed area tables* (SATs), we propose a GPU reduction scheme that can efficiently perform the visibility estimation for multiresolution data blocks, ensuring real-time update of view-dependent information and interactive LOD selection.

The remainder of the paper is structured as follows. First, we present background and review related work in Section II. In Section III, we briefly introduce our multiresolution data representation for large three-dimensional data sets. In Section IV, we describe our multiresolution LOD selection and rendering algorithm in detail. Experimental evidence showing the visual quality gain using our image-based LOD selection over data-based ones is provided in Section V. The paper is concluded in Section VI with future work for our research.

II. Background and Related Work

A. Background

Multiresolution Data Representation: Building a multiresolution data hierarchy allows the user to visualize data at different scales, and balance image quality and computation speed. A number of techniques have been developed to provide hierarchical data representation for volumetric data, such as the *Laplacian pyramid* [9], multi-dimensional trees [32], and octree-based hierarchies [1], [17]. Muraki [25] first introduced the use of wavelet transforms for volumetric data. Westermann [31] presented a framework for approximating the volume rendering integral using multiresolution spaces and wavelet projections. More recently, Guthe et al. [13] presented a wavelet-encoded hierarchical representation for large volume data sets that supports interactive walkthroughs on a single commodity PC.

Image-Based Quality Measurement: The lack of correlation between the type of error in an image and the response of the *human visual system* (HVS) to different types of errors prompted researchers to develop image-based metrics. Jacobs et al. [15] introduced an image-query metric for searching in an image database using a query image similar to the intended target. The metric makes use of multiresolution wavelet decompositions of the query and database images, and operates on the coefficients of these decompositions. Gaddipati et al. [7] presented a wavelet-based metric which captures the change in images wrought by operators and the image synthesis algorithms. Sahasrabudhe et al. [27] proposed a quantitative technique which accentuates differences in images and data sets through a collection of partial metrics. A study of different image comparison metrics, categorized into spatial domain, spatial-frequency domain, and perceptually-based metrics, was presented in [33]. Alternatively, Wang et al. [30] proposed the use of structural similarity for the design of image quality measures. Experimental results show that their *Structural Similarity Index* simulates the response of the HVS with low computation cost.

In the context of large volume visualization, an image-based metric is difficult to incorporate because of the following reasons: First, image-based metrics need to consider run-time information, such as the viewing, projection, and occlusion. Unlike most data-based metrics which can be easily computed in a preprocessing stage, to get view-dependent occlusion information for a large data set, one has to resort to either sophisticated precomputation with considerable overhead [8], or run-time calculation with rough approximation [12]. Next, the user may adjust the transfer function during the rendering in order to reveal different features. Image-based metrics should be adaptive to run-time transfer function changes. Previous work on large data visualization usually assumes the input transfer function is fixed, or is limited to a family of transfer functions consisting of a series of basis functions [8]. Last, the human observer plays a central role in perceiving the image quality. Therefore, image-based metrics should also take into account human perception [28] in the visualization process. The factors need to be considered include the perception of distance, coverage, shape, color, occlusion, texture, and lighting. In this paper, we integrate an image-based quality metric into the multiresolution LOD selection and rendering framework.

Visibility Computation: To accelerate the process of image generation, visibility culling has long been employed [2] in rendering large polygonal models as well as volumetric data sets. Klosowski and Silva [16] introduced the time-critical *Prioritized-Layered Projection* (PLP) algorithm for fast rendering of high depth complexity scenes, using a solidity-based metric for visibility estimation. A similar approach that

TABLE I
THE COMPARISON OF THE THREE APPROACHES.

approach	Guthe et al. [12]	Ljung et al. [22]	our approach
data hierarchy	octree-based	flat block-based	octree-based
error evaluation	RGB, use max error	CIELUV, MSE-based	CIELUV, image quality measure
block projection	projection size	not considered	luminance, projection size, thickness
visibility estimation	assume uniform opacity occlusion only raycasting, CPU	use simplified histogram occlusion only raycasting, CPU	use low resolution data emission + occlusion SAT, GPU
transfer function	not adaptive	adaptive, 12-level simplified histogram	adaptive, 256-level histogram

integrates occlusion culling with view-dependent rendering was given in [4]. Gao et al. [8] proposed a *Plenoptic Opacity Function* (POF) scheme, which encodes the view-dependent opacity of a volume block for visibility culling of large volume data. Utilizing visibility information for multiresolution volume rendering has not been widely studied, nor has the potential of using programmable graphics hardware for visibility estimation been fully explored.

B. Related Work

Two recent methods have been proposed for error measurement and visibility calculation within the multiresolution volume rendering framework. Guthe et al. [12] presented several improvements on compression based multiresolution rendering of large data sets to speed up the volume rendering process. The screen-space error was measured as the maximum error produced by each block multiplied by its projection size. The maximum error was calculated by simply adding the differences of RGB color and opacity components, rather than more correct opacity-modulated color differences. The visibility estimation was performed for empty space skipping and occlusion culling to speed up the rendering. They took a conservative way of visibility testing, and assumed a uniform opacity for each data block for very rough approximation. Ljung et al. [22] focused on incorporating transfer function into adaptive decompression of volume data for multiresolution volume rendering. Similar to [21], they took a flat block-based volume decomposition approach. At the compression stage, they calculated the meta-data for each block: the average scalar value, the *root-mean-square* (RMS) wavelet coefficients, and a simplified histogram. The error metric was based on a simplified version of MSE in the CIELUV space. To account for occlusion, a low resolution ray-casting renderer was used to estimate the average opacity of each block, followed by empirical tests for approximating the simplified discrete rendering equation with no emission factor.

In our work, the goal is to incorporate an image-based quality metric for multiresolution volume rendering of large data sets. Thus, our main focus is on quality-driven interactive LOD selection, rather than compression based rendering [12], or transfer function based decompression [22]. To achieve this goal, we propose much more refined solutions at each step of our algorithm. Our image-based quality metric takes into account the emission as well as the occlusion of the multiresolution data blocks, and is more accurate than the simplified ones in [12] and [22]. We present a summary table scheme to account for the run-time transfer function change with much higher precision (256-level histogram) than the one (12-level simplified histogram) in [22]. Using simplified histogram has the risk of missing important details in the data. Therefore, our refined solution is more suitable for large data sets with high dynamic range. Our scheme allows one to update the errors for a large volume of size around 1024^3 within seconds. Also, we introduce a GPU-based reduction scheme for getting estimated visibility for the data blocks in real time, while both of those methods in [12] and [22] used only the software raycasting approach. We use low resolution data for visibility estimation, which is more exact than just assuming a uniform opacity [12] or taking the simplified histogram [22] for each block. In Table I, we list the major differences between our approach and the two related ones.

III. MULTIRESOLUTION DATA HIERARCHY

To build a multiresolution data hierarchy from a large three-dimensional data set, we use wavelet transforms to convert the data into a hierarchical multiresolution representation, called the *wavelet tree* [13]. The wavelet tree construction algorithm starts with subdividing the original 3D volume into a sequence of blocks with the same size (assuming each has N voxels). These raw volume blocks form the leaf nodes of the wavelet tree. After performing a 3D wavelet transform on each block, a low-pass filtered subblock of size $N/8$ and wavelet coefficients of size $7N/8$ are produced. The low-pass filtered subblocks from eight adjacent leaf nodes in the wavelet tree are grouped into a single block of N voxels, which becomes the low resolution data stored in the parent node. We recursively apply this 3D wavelet transform and subblock grouping process in a bottom-up manner till the root of the tree is reached, where a single block of size N is used to represent the entire volume. To reduce the size of the coefficients stored in the wavelet tree, the wavelet coefficients in each tree node are set to zero if they are smaller than a user-specified threshold. These wavelet coefficients are then compressed using run-length encoding combined with a fixed Huffman encoder. Note that in the wavelet tree, the multiresolution data blocks associated with all the tree nodes have data of the same size, which is N. However, the spatial resolutions they represent may vary, depending on which level of the tree the corresponding nodes lie on.

Coupled with the construction of the wavelet tree, multiresolution error ε is evaluated for each of the tree nodes. We calculate the error as the summation of the errors between the parent block and its eight immediate child blocks. We also

take into account the maximum error of the child blocks, as an approximation of the error between the parent block and the original full-resolution data block it represents. Written in equation:

$$\varepsilon_{b_i} = \sum_{j=0}^{7} \varepsilon_{ij} + \max\{\varepsilon_{b_j}|_{j=0}^{7}\} \quad (1)$$

where ε_{ij} is the voxel-wise error between parent block b_i and its jth child block b_j (Note that b_j contributes one eighth of the low-pass filtered data to b_i. For each voxel value in b_j, we linearly interpolate its corresponding value from its low-pass filtered data in b_i.); ε_{b_i} and ε_{b_j} are the multiresolution errors of blocks b_i and b_j respectively. As a special case, if block b_i is associated with a leaf node in the hierarchy, we define its error as a small constant C. Depending on the need, ε_{ij} can be calculated in different ways. For example, we can directly calculate it in the scalar data space using MSE or SNRMSE (MSE of SNR), or in the RGB or the CIELUV color space. The multiresolution errors in the data hierarchy are then normalized for our use.

IV. LOD Selection and Rendering Algorithm

Our multiresolution LOD selection and rendering algorithm optimizes the quality of rendered images through the use of an image-based quality metric. Our quality metric evaluates the importance values of multiresolution data blocks by examining the contribution of data blocks to the final image, based on the *discretized volume rendering integral* (DVRI). The evaluation approximates the emission of each block, as well as takes into account the occlusion caused by the blocks in front of it. To capture the multiresolution error in the data hierarchy, we modulate the importance value with the distortion between low and high resolution data blocks, calculated in the roughly perceptually-uniform CIE $L^*u^*v^*$ (CIELUV) color space. To ensure a real-time update of the quality metric, we propose a summary table scheme to respond to changes of the transfer function, and a GPU reduction scheme for visibility estimation. At run time, for a given viewing direction, the LOD selection is made based on a priority queue scheme utilizing the importance values of multiresolution blocks as their priority values. The wavelet tree traversal maintains the LOD as a cut through the hierarchy, and the importance values dictate the sequence of LOD refinements. A certain number of blocks up to a user-specified budget are extracted and sent to the texture hardware for rendering.

A. Volume Rendering Integral

According to the emission-absorption optical model [23], the *volume rendering integral* (VRI) that calculates the amount of light I along a viewing ray r through the volume is given by:

$$I_r = \int_0^D \tilde{c}(s(\vec{x}(\lambda)))\ \exp(-\int_0^\lambda \tau(s(\vec{x}(\lambda')))\ d\lambda')\ d\lambda \quad (2)$$

where $s(\vec{x}(\lambda))$ is the scalar value at position $\vec{x}(\lambda)$ in the volume, parameterized by the distance λ to the viewpoint; $\tilde{c}(s)$ is the volume source term or intensity; $\tau(s)$ defines the attenuation function.

In general, the VRI cannot be evaluated analytically. Therefore, practical volume rendering algorithms discretize the VRI by numerical approximation. Using Riemann sum for n equal ray segments of length D/n, and further approximating the exponential function with the first two terms of the Taylor series expansion, we get the *discretized volume rendering integral* (DVRI) [24], also known as the compositing equation [20]:

$$I_r = \sum_{i=0}^{n} c(s_i)\alpha(s_i)\prod_{j=0}^{i-1}(1-\alpha(s_j)) \quad (3)$$

where $c(s)$ and $\alpha(s)$ define the color and opacity transfer function. This equation denotes that at each discrete sample position i along the viewing ray r in the volume, light is emitted according to the term $c(s_i)\alpha(s_i)$, which is absorbed by the volume at all positions along r in front of i according to the term $\alpha(s_j)$. Eqn. 3 serves as the foundation for our design of importance values for multiresolution data blocks.

B. Importance Value Design

The DVRI in Eqn. 3 evaluates the amount of light visible to the eye on a per-ray basis. It is also possible to look at the equation on a per-slice basis, which leads to the popular slice-based compositing technique for volume rendering. In this paper, the underlying entity for our LOD selection and rendering algorithm is a data block. Therefore, we evaluate the importance values of multiresolution data blocks by approximating Eqn. 3 on a per-block basis. The importance value of a data block b along the viewing direction r is calculated as follows:

$$I_b = (c(\mu)\alpha(\mu)\ \cdot\ t\ \cdot\ a)\ \cdot\ v \quad (4)$$

where μ is the mean scalar data value of block b; $c(\mu)$ and $\alpha(\mu)$ define the color and opacity transfer function (we actually calculate the magnitude of its corresponding CIELUV color); t is the average thickness (the length of the viewing ray segment inside the block) of block b; a is the screen projection area of the block, and v is its estimated visibility. Similar to Eqn. 3, here $((c(\mu)\alpha(\mu)\cdot t\cdot a)$ approximates the emission of block b along direction r, and v accounts for the attenuation. Given a viewing direction r, I_b essentially evaluates the contribution of block b to the final image.

If we record the mean scalar value μ of each block during the construction of the multiresolution data hierarchy, we can quickly compute $c(\mu)$ and $\alpha(\mu)$ on the fly. Also, given a viewing direction r, the average thickness t and projection area a of a block can be easily calculated at run time. However, to obtain the estimated visibility v of a block interactively is non-trivial, and we will describe our real-time GPU-based solution in Section IV-E.

Even if two multiresolution data blocks have the same approximate emission and absorption terms, the distortions between the blocks and their children can be different. Taking

into account the relative distortion, we modulate the importance value with the multiresolution error between low and high resolution data blocks. Eqn. 4 becomes:

$$I_b = (c(\mu)\alpha(\mu) \cdot \varepsilon \cdot t \cdot a) \cdot v \tag{5}$$

where ε is the distortion between block b and its higher resolution child blocks, normalized to $[0,1]$. The motivation behind this modulation is that if a block contains larger distortion, then it should receive a higher priority value for LOD refinements.

C. Multiresolution Error Evaluation

Previously, researchers have proposed various ways to calculate the multiresolution error ε in the scalar data space [1], [19], [29], and in the RGB [5], [12] or the CIELUV [22] color space. In this paper, we take an image-space approach and opt to evaluate the multiresolution error in the perceptually-adapted CIELUV color space, as suggested by Glassner [10].

Let us consider two data blocks b_i and b_j, where b_j is an immediate child block of b_i. We define the multiresolution error between b_i and b_j as follows:

$$\varepsilon_{ij} = \tilde{\sigma}_{ij} \cdot \frac{\tilde{\mu}_i^2 + \tilde{\mu}_j^2 + C_1}{2\tilde{\mu}_i\tilde{\mu}_j + C_1} \cdot \frac{\tilde{\sigma}_i^2 + \tilde{\sigma}_j^2 + C_2}{2\tilde{\sigma}_i\tilde{\sigma}_j + C_2} \tag{6}$$

where $\tilde{\sigma}_{ij}$ is the covariance between b_i and b_j; $\tilde{\mu}_i$ and $\tilde{\mu}_j$ are the mean values of b_i and b_j respectively; $\tilde{\sigma}_i$ and $\tilde{\sigma}_j$ are the standard deviations of b_i and b_j respectively (small constants C_1 and C_2 are included to avoid instability when $\tilde{\mu}_i\tilde{\mu}_j$ and $\tilde{\sigma}_i\tilde{\sigma}_j$ are very close to zero):

$$\tilde{\sigma}_{ij} = \frac{1}{N-1}\sum_{k=1}^{N}(\tilde{x}_{ik} - \tilde{\mu}_i)(\tilde{x}_{jk} - \tilde{\mu}_j) \ ; \tag{7}$$

$$\tilde{\sigma}_i = \frac{1}{N-1}\sum_{k=1}^{N}(\tilde{x}_{ik} - \tilde{\mu}_i)^2 \ ; \quad \tilde{\sigma}_j = \frac{1}{N-1}\sum_{k=1}^{N}(\tilde{x}_{jk} - \tilde{\mu}_j)^2 \ . \tag{8}$$

Here, N is the number of voxels in the block, and $\tilde{x}$ is the volume source term. Using Eqn. 1 and 6, we can calculate the multiresolution error for each tree node as we build up the multiresolution data hierarchy. Eqn. 6 consists of three parts, namely, *covariance*, *luminance distortion*, and *contrast distortion*. The first part is the covariance between b_i and b_j, which measures the degree of linear correlation between the two blocks ($\tilde{\sigma}_{ij}$ is always non-negative since we actually calculate it based on the CIELUV color differences of the pairs $(\tilde{x}_{ik}, \tilde{\mu}_i)$ and $(\tilde{x}_{jk}, \tilde{\mu}_j)$, as explained in Eqn. 9 and 10). Even though b_i and b_j are linearly related, there still might be relative distortions between them. Therefore, we add two more parts to the equation. The second one, measures how close the mean luminance is between b_i and b_j. The minimum value of 1.0 is achieved if and only if $\tilde{\mu}_i = \tilde{\mu}_j$. On the other hand, $\tilde{\sigma}_i$ and $\tilde{\sigma}_j$ can be viewed as estimate of the contrast of b_i and b_j, so the third part measures how similar the contrasts of the two blocks are. Also, the minimum value of 1.0 is achieved if and only if $\tilde{\sigma}_i = \tilde{\sigma}_j$. Collectively, these three parts capture the distortion between the two blocks. The luminance distortion and contrast distortion are originally from the image quality assessment literature [30], and have been shown to be consistent with the luminance masking and contrast masking features in the HVS respectively.

One should notice that the input source terms, $\tilde{x}$ and $\tilde{\mu}$, are CIELUV color values, rather than original scalar data values. Accordingly, we define $\tilde{x}_{ik} - \tilde{\mu}_i$ as follows ($\tilde{x}_{jk} - \tilde{\mu}_j$ can be defined similarly):

$$\tilde{x}_{ik} - \tilde{\mu}_i = \Delta E(f(c_{rgb}(x_{ik})\alpha(x_{ik})), \ f(c_{rgb}(\mu_i)\alpha(\mu_i))) \tag{9}$$

where x_{ik} is the scalar data value at the kth voxel position in block b_i; μ_i is the mean scalar value of b_i; c_{rgb} and α define the color and opacity transfer function; f is the function that converts an RGB color to its CIELUV color [6]; ΔE is the Euclidean distance between a pair of colors specified in the CIELUV color space:

$$\Delta E = \sqrt{\Delta L^{*2} + \Delta u^{*2} + \Delta v^{*2}} \tag{10}$$

where ΔL^*, Δu^*, and Δv^* are the differences of L^*, u^*, and v^* components for the pair of CIELUV colors.

D. Summary Table Scheme

As we can see, the calculation of multiresolution error ε_{ij} in Eqn. 6 requires the input of the color and opacity transfer function (Eqn. 9). At run time, whenever the user adjusts the transfer function, the multiresolution errors in the entire data hierarchy have to be recomputed all over again. This entails a considerable amount of computation overhead and makes the whole LOD selection and rendering process non-interactive. In the following, we describe a summary table scheme that ensures real-time update of the errors in response to transfer function changes.

Our summary table scheme is based on the observation that, for large data sets, the size of the data range is often many orders of magnitude smaller than the number of voxels in the volume. For instance, the RMI data set is byte (8-bit) data with a data range size of 256. However, the number of voxels in the volume is $2048 \times 2048 \times 1920$. Therefore, instead of calculating Eqn. 7 and 8 by going through the individual voxels, it suffices to count the frequencies of unique error terms, which is much faster (similar observations have been made and utilized in [5], [18]). In the case of byte data, there are $256^2 = 65536$ combinations for $\tilde{\sigma}_{ij}$, and only 256 cases for $\tilde{\sigma}_i$ or $\tilde{\sigma}_j$. To compute the error, rather than adding individual error terms voxel by voxel, we add the products of each unique error term and the frequency of that term.

To realize this, first of all, for each of the data blocks at the multiresolution hierarchy, we precompute the mean scalar value μ, and keep a local *histogram table* $\mathbb{H}$ (256 entries):

$$m = \mathbb{H}\ (x_i)$$

where x_i is the scalar value, m is the frequency of x_i in the block.

Next, for each data block associated with a non-leaf node in the hierarchy, we keep a local *correspondence table* $\mathbb{C}$ (65536 entries):

$$m = \mathbb{C}\ (x_i,\ x_j)$$

where x_i is the scalar data value in the current (parent) block; x_j is the data value in one if its eight immediate child blocks; m is the frequency of the data pair. We refer to these histogram and correspondence tables as *summary tables*. They are created during the construction of the data hierarchy and are precomputed only once. Besides this, we keep a global *distance table* $\mathbb{D}$ ($1+2+\ldots+255 = 32640$ entries):

$$\Delta E = \mathbb{D}\ (x_i,\ x_j)$$

where x_i and x_j are scalar data values, and $x_i < x_j$; ΔE is the distance between x_i and x_j in the CIELUV color space.

Finally, we keep a global *function table* $\mathbb{F}$ (the number of entries in the transfer function, usually 256):

$$L^*u^*v^* = \mathbb{F}\ (rgb\alpha)$$

where $rgb\alpha$ is the RGB color and opacity in the current transfer function, $L^*u^*v^*$ is the corresponding CIELUV color. The global distance table and function table are initialized at run time and are updated when the transfer function changes.

At run time, we can quickly calculate the multiresolution error ε for each block using Eqn. 1 and 6-10, by looking up the mean scalar value μ and summary tables ($\mathbb{H}$ and $\mathbb{C}$) stored in each of the blocks, as well as the global distance table $\mathbb{D}$ and function table $\mathbb{F}$. The lookup relationships are as follows:

$$\begin{aligned} \tilde{\mu}_i, \tilde{\mu}_j &\leftarrow \mu, \mathbb{F}\ ; \\ \tilde{\sigma}_i, \tilde{\sigma}_j &\leftarrow \mathbb{H}, \mathbb{F}\ ; \\ \tilde{\sigma}_{ij} &\leftarrow \mathbb{C}, \mathbb{D}\ . \end{aligned}$$

Whenever the user changes the transfer function, only the global distance table and function table need update.

For data sets other than byte data, quantization is necessary in order to reduce the size of summary tables (otherwise, the size of these tables could be even larger than the size of actual data blocks, and the time for error calculation would increase dramatically). For example, we can quantize the scalar data range into 256 levels either uniformly or based on the histogram of the whole data set. In this way, the total size of the summary tables will remain small regardless of the data type of the input volume.

One can observe that, usually there is a strong degree of correlation between parent and child blocks in the data hierarchy. This means that in the correspondence table $\mathbb{C}$, when x_i is close to x_j (i.e., the entry is close to the major diagonal of the table), the frequency m is large. m is smaller, actually often zero, if the entry is further away from the major diagonal. Leveraging this observation, we can perform run-length encoding on the correspondence table $\mathbb{C}$ in a zigzag manner, as illustrated in Fig. 2. The zigzag run-length encoding not only reduces the storage of correspondence tables, but also improves the run-time performance. For instance, using the run-length encoded correspondence tables $\mathbb{C}$ for the RMI data set, the total size of summary tables reduces from 208MB to 44.1MB, and accordingly, the time to update multiresolution errors decreases from 43 seconds to 13 seconds.

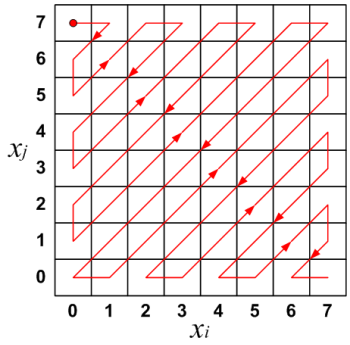


Fig. 2. Run-length encoding on the correspondence table $\mathbb{C}$ in a zigzag manner. An example of an 8×8 table is shown here. The encoding starts from the red circle, and follows the red arrows.

E. GPU-Based Visibility Estimation

Obtaining the exact visibility of the multiresolution data blocks requires rendering the blocks. This is similar to rendering the entire hierarchy, which could be rather slow and defeats the purpose of the visibility test. For coarse-grained multiresolution rendering, getting an approximate visibility of a block suffices. In this scenario, the visibility computation should be done prior to the actual rendering of blocks.

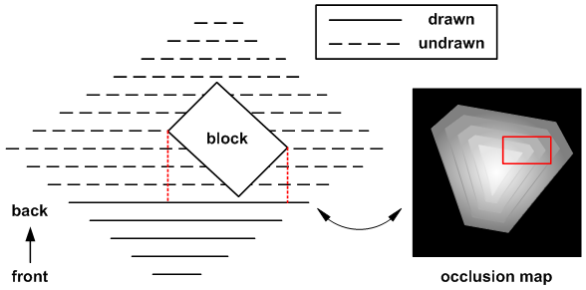


Fig. 3. Visibility estimation via rendering a low resolution of the data. The visibility of a block is acquired when its nearest vertex is in-between the current slice and the latest drawn one.

In our algorithm, we render a low resolution of the data (for example, we can use the root of the data hierarchy) by drawing front-to-back view-aligned slices, and evaluate the approximate visibility of all the blocks during the slice drawing, as illustrated in Fig. 3. The visibility of a block is computed as $(1-\alpha)$, where α is the *average* opacity within the block's screen projection on the occlusion map, accumulated right before the first slice intersecting the block (α is considered as the accumulated opacity in front of that block). Here we assume the visibility of a data block is independent of the resolutions of all the occluding blocks in front of it, because opacity correction is performed to compensate the varying slice distances within data blocks of different resolutions. Note that a conservative way of taking the *minimum* opacity, commonly used in occlusion culling, is unnecessary. For occlusion culling, the decision is to either

render or discard a block, and getting the minimum opacity is crucial to avoid producing incorrect images by leaving holes. For multiresolution rendering, the whole volume is rendered anyway, because the question is to select proper LODs for different regions within the volume, rather than to render or discard a region. Therefore, it is reasonable to get the average instead of the minimum opacity.

To compute the estimated visibility for a data block, a naive way is to read the alpha channel of the framebuffer to an off-screen buffer after a certain number of slices are rendered, and iterate through the pixels that the data block projects to and obtain an average of the opacities. This software approach is slow due to the framebuffer reads from the GPU to the CPU (refer to the timing in Table IV). The testing time is proportional to the size of output images and the number of pixels each block projects to. To minimize the transferring of pixels from the GPU to the CPU, we move all operations to the GPU. Our GPU reduction scheme is based on the *summed area tables* (SATs) [3]. The construction of a SAT is linear to the number of pixels on the area being considered, in our case the whole rendering screen. However, it only takes constant time to retrieve the sum over any rectangular area, which is done in one addition and two subtractions. This fits perfectly in getting the corresponding averages from the projections of the blocks. We build the SATs in multiple passes with the support of framebuffer objects having double auxiliary buffers (see the Appendix for the implementation detail). Getting the estimated visibility for a block is performed by another fragment program that looks up the four corners of its projection in the output auxiliary buffer holding the SAT.

Testing shows that the time to perform GPU-based visibility estimation is not negligible. For instance, with output image resolution of 512^2, each SAT takes around 10ms to compute on an nVidia GeForce 7800 GT graphics card. In total, it took about 0.3 second (recall that we need to recompute the SAT whenever a certain number of slices of the low resolution data are drawn) to update the visibility of 10499 non-empty blocks for the RMI data set. If we perform such a test for every frame, then the frame rate would be limited to around 3.3fps. To overcome this constraint, we incorporate the following two strategies to improve the rendering frame rates.

First, the number of block budget the user specifies is usually much smaller than the total number of blocks in the data hierarchy. For such a typical block budget and a given transfer function, normally a large portion of the updated visibility of blocks farther away from the viewpoint (more likely to be occluded from the blocks in front of them) never gets a chance to contribute to the current LOD decision. Actually, for the RMI data set, tests show that about 30-50% of the total number of blocks fall into this category. In view of this, we can only draw the front slices up to a certain percentage of the total number of slices, and update the corresponding visibility of blocks that are closer to the viewpoint. Any block whose visibility is not updated in this run uses whatever it has from the latest previous run. In this way, we can reduce the visibility estimation time to around 0.18 second for the RMI data set, if we only update 60% of the front slices and blocks.

Second, the visibility of the blocks only changes a little bit if the view does not change greatly. Therefore, if the angle between the current viewing direction and the latest one with the visibility updated is less than a threshold angle θ, we do not update the visibility and use whatever we have from the latest run. Otherwise, we need to update again. Here, θ is a predefined small angle (initialized to 5 degrees in our test), and is adaptive to the zooming of the data during the rendering.

By reducing the load to perform each run of visibility estimation and the frequency of performing such estimations, we can reuse visibility computation and utilize frame to frame coherence, achieving smoother rendering and better frame rates.

F. LOD Selection and Rendering

At run time, the user specifies the number of blocks as a budget for rendering. Given a viewing direction, the LOD selection is made based on a priority queue scheme. The priority values of blocks are their importance values calculated according to Eqn. 5 (where v is updated per view and ε per transfer function). Thus, a block with a higher importance value is more likely to be selected for refinement during the wavelet tree traversal. Constrained by the given budget, the traversal maintains the LOD as a cut through the multiresolution data hierarchy, and refines the blocks on the cut in a greedy manner.

The LOD selection and rendering works as follows: First, we initialize the priority queue with the data block of the lowest resolution, i.e., the root of the multiresolution data hierarchy. Then, we successively refine the block with the highest priority value in the queue until the budget is met. The refinement is performed by deleting the block b with the highest priority value, updating the importance values of b's eight child blocks, and inserting the child blocks into the queue. Finally, all the data blocks in the queue are sorted in front-to-back viewing order. These blocks are reconstructed, if necessary, and sent to texture hardware for rendering.

As we may anticipate reusing most of the reconstructed data blocks for subsequent frames due to the spatial locality and coherence exploited by the multiresolution data hierarchy, it is desirable to cache the data blocks that have already been reconstructed for better performance. The user can predefine a fixed amount of disk space and memory dedicated for the caching purpose. Upon requesting a data block for the rendering, we retrieve its data from memory, provided the block is cached in main memory. Otherwise, we need to load the data from disk if the reconstructed data block is cached on disk. If it is neither cached in memory nor on disk, then we need to reconstruct the data block and load it into main memory. When the system runs short of the available storage for caching the reconstructed blocks, our replacement scheme will swap out a data block that has been visited least often.

V. Results and Discussion

A. Results

We experimented with our LOD selection and rendering algorithm on the VisWoman and RMI data sets, as listed in

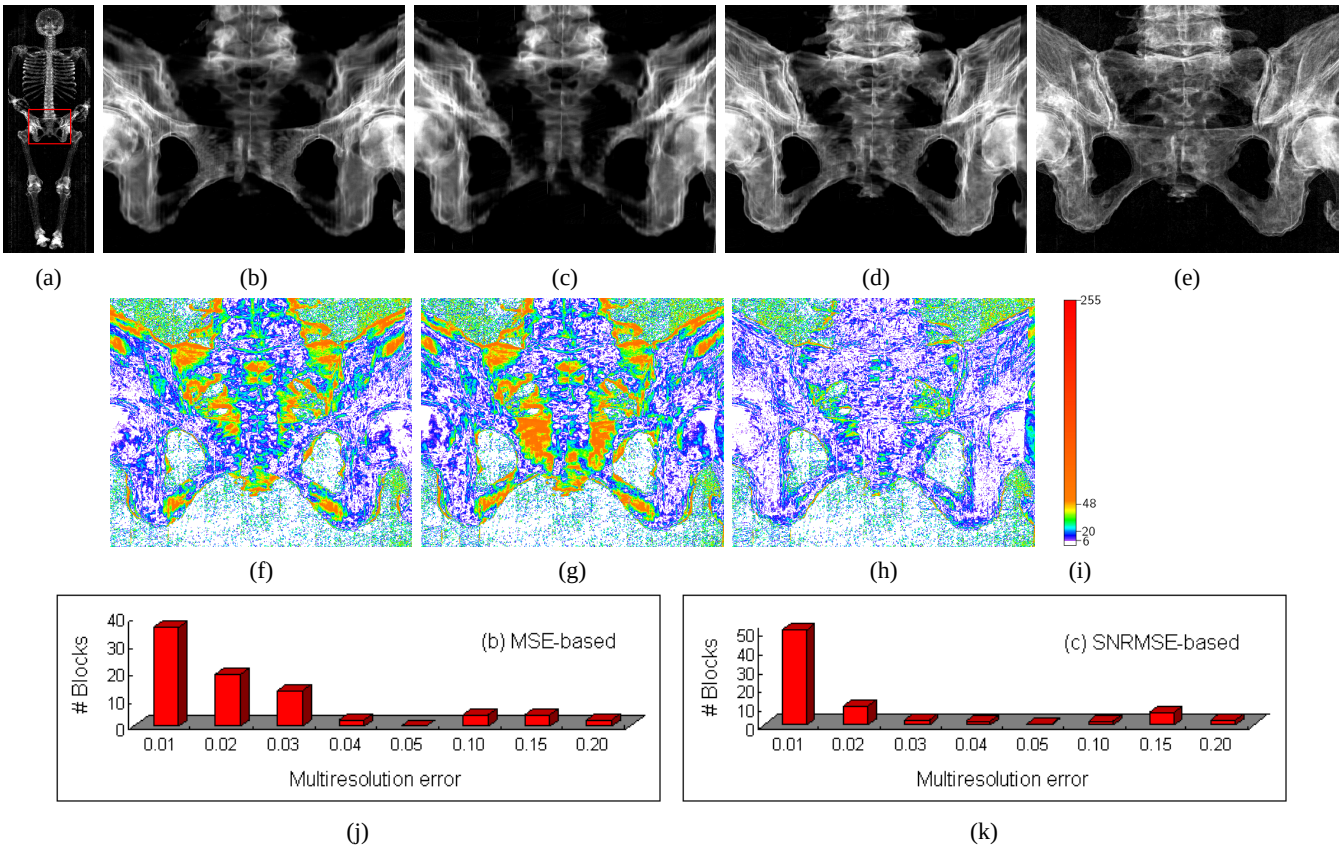


Fig. 4. First row: (a) shows an overview and (b)-(e) show a zoom to the pelvis. One can observe that (d) (image-based, 77 blocks, 8.29% of full data) shows more details than (b) (MSE-based, 80 blocks, 8.61%) and (c) (SNRMSE-based, 79 blocks, 8.50%). The reference image (e) is rendered with full resolution (929 blocks). Second row: objective image comparison in the CIELUV color space. (f), (g), and (h) show the difference between (b) and (e), (c) and (e), and (d) and (e) respectively. The color map (i) maps ΔE to color. Third row: (j) and (k) show the numbers of blocks selected in each of the multiresolution error levels for (b) and (c) respectively.

Table II. The decision for the block size was based on the cost of performing the wavelet transform for a single data block, and the rendering overhead for final image generation. We extended one voxel overlapping boundaries between neighboring blocks in each dimension when breaking the original volume data into blocks in order to produce seamless rendering. As a result, both hierarchies have a tree depth of six. For both data sets, the Haar wavelet transform with a lifting scheme was used to construct the data hierarchies. A *lossless* compression scheme was used with the threshold set to zero to compress the wavelet coefficients. For LOD rendering, we compared the images generated using our image-based quality metric and two data-based metrics: MSE and SNRMSE (MSE of SNR). For the MSE-based (SNRMSE-based) metric, we directly used the multiresolution error as the importance value, while ε_{ij} is the MSE (SNRMSE) of the scalar data values of blocks b_i and b_j in Eqn. 1. Similar block budgets were set for all three cases for fair comparison. All tests were performed on a 3.0GHz Intel Xeon processor with 3GB main memory, and an nVidia GeForce 7800 GT graphics card with 256MB video memory.

The first row of Fig. 4 shows the LOD rendering of the VisWoman data set using the three metrics. The full-resolution reference image is provided for comparison. We used a transfer function that highlights the skeleton. It can be observed that

TABLE II
THE VISWOMAN AND RMI DATA SETS.

data set (type)	VisWoman (short)	RMI (byte)
volume dimension	$512 \times 512 \times 1728$	$2048 \times 2048 \times 1920$
block dimension	$32 \times 32 \times 64$	$128 \times 128 \times 64$
volume (block) size	864MB (128KB)	7.5GB (1MB)
# non-empty blocks	9446	10499
compression ratio	2.37:1	5.60:1

when we rendered the data in low resolution, the LOD selection using the image-based quality metric shows more details than the MSE-based and SNRMSE-based metrics. In Fig. 4 (j) and (k), we compare the numbers of blocks selected in each of the multiresolution error levels for (b) and (c) respectively (the multiresolution errors have been normalized). Assuming that our multiresolution errors are able to capture the structural distortion of the data, we can infer that both MSE-based and SNRMSE-based metrics perform much worse due to their selections of not-so-highly prioritized multiresolution data blocks. An objective image comparison was also conducted to testify the visual quality gain obtained using our image-based quality metric. We calculated the pixel-wise differences between the low resolution image and the reference image in the CIELUV color space. The difference threshold $\Delta E \geq 6.0$

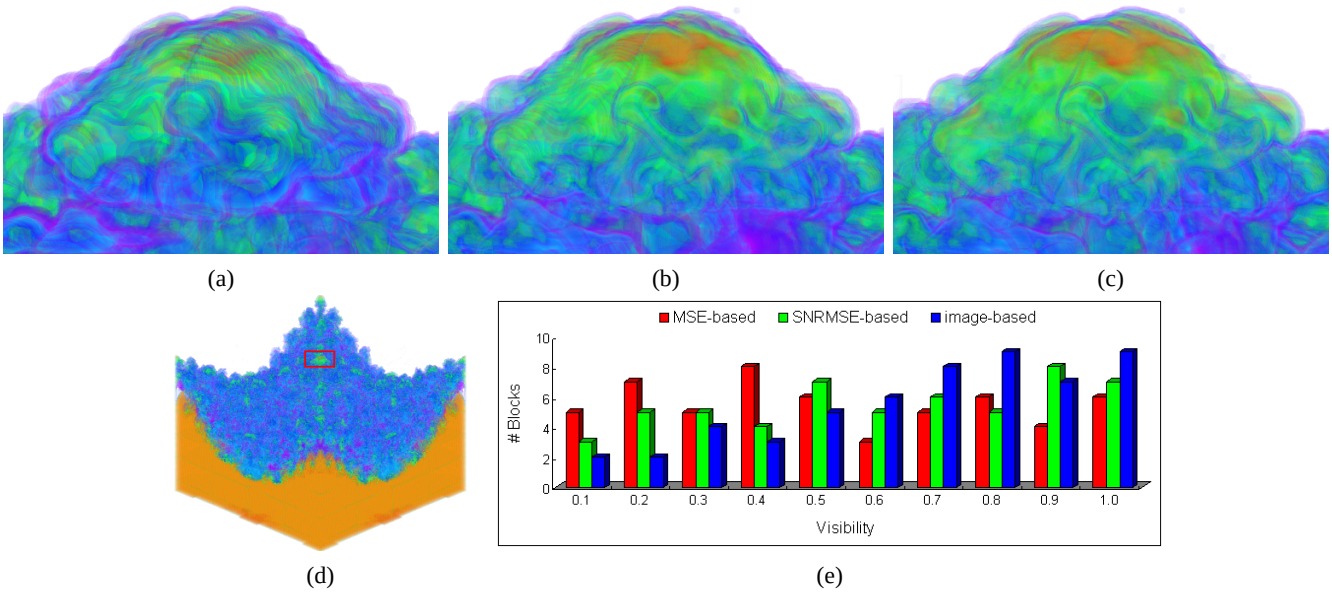


Fig. 5. (a) (MSE-based, 55 blocks), (b) (SNRMSE-based, 55 blocks), and (c) (image-based, 55 blocks) show a zoom to the center of the RMI data set, while an overview is shown in (d). (e) shows the numbers of blocks rendered in each of the ten visibility levels for (a)-(c) respectively.

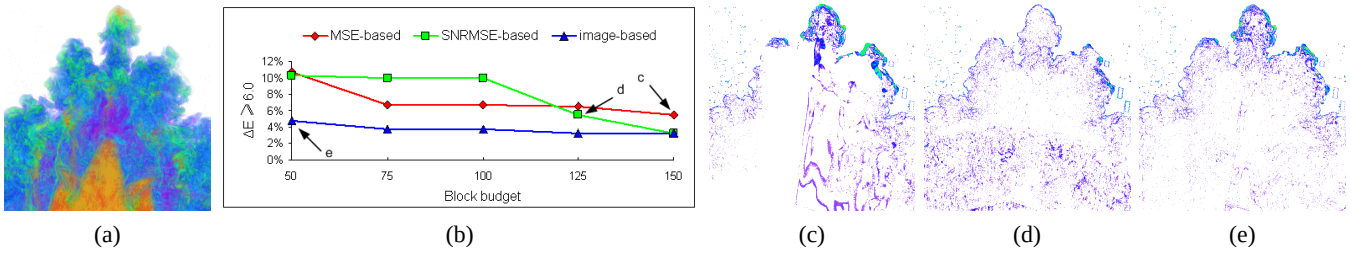
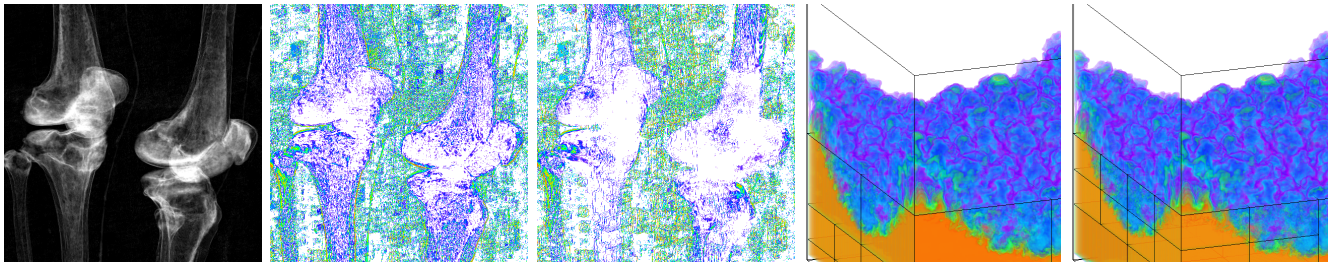


Fig. 6. (a) shows the reference image with full resolution (1237 blocks). (b) shows the percentage of pixels with $\Delta E \geq 6.0$ in the difference images for the three metrics, under five different block budgets. (c)-(e) show three difference images near 5%, as indicated in (b): (c) (MSE-based, 150 blocks), 5.51%, (d) (SNRMSE-based, 128 blocks), 5.48%, and (e) (image-based, 50 blocks), 4.75%.

gives the noticeable pixel distortion [22]. At the second row of Fig. 4, we show these difference images side by side. Clearly, the ones with the MSE-based and SNRMSE-based metrics contain larger visual distortion. Another rendering example of the VisWoman data set is shown in Fig. 1. We can see that the image-based LOD selection shows clearer structures along the spine.

Fig. 5 shows the LOD selection and rendering of the RMI data set using the three metrics. We zoomed into the center of the data and compared fine details after an overview. The image-based quality metric takes into account the multiresolution error and visibility of each data block, thus puts more refinement effort on the blocks that have larger visual contribution. Fig. 5 (e) shows the numbers of blocks rendered in each of the ten visibility levels for Fig. 5 (a)-(c) respectively. As we can see, compared with the ones with MSE and SNRMSE, the image-based one selected more blocks with higher visibility. Similar conclusions can be drawn from Fig. 6, where the image-based quality metric achieves near 5% noticeable pixel distortion with a block budget of only 50, as opposed to 150 and 125 for the MSE-based and SNRMSE-based ones respectively. To verify that including estimated visibility v in Eqn. 5 does help in LOD selections, we tested our image-based LOD selection algorithm without and with visibility information. The results in Fig. 7 show that adding visibility information in the LOD selection leads to more refinement on blocks closer to the viewpoint and with higher visibility. All the results in Fig. 1 and 4-7 confirm the effectiveness of our image-based LOD selection algorithm.

We also experimented with our summary table scheme for updating the multiresolution errors. With 256-level histograms and transfer functions, the statistics is shown in Table III. For both data sets, the zigzag run-length encoding scheme takes at least 70% less time to update the multiresolution errors with much smaller storage overhead than the one with no coding. The summary table scheme proved very efficient in response to the transfer function changes with negligible storage overhead. A rendering example of the VisWoman data set is shown in Fig. 8, where a different transfer function was used to highlight both the skin and the skeleton. We zoomed into the left foot and rendered in close to full resolution. The three methods generated similar results as we approached full resolution. Still, it can be seen that the MSE-based method contains much noise coming from the 3D test bed surrounding the cadaver

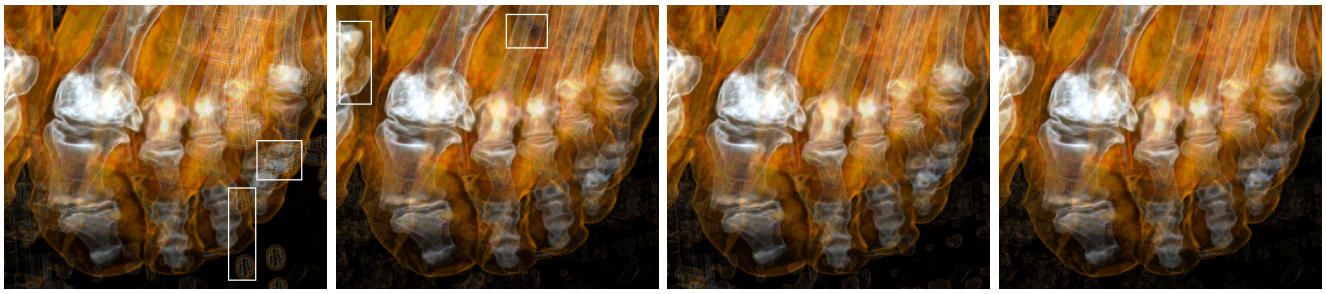


(a) (b) (c) (d) (e)

Fig. 7. (a) shows a zoom to the knee joints of the VisWoman data set, rendered with full resolution (1033 blocks). (b) (w/o visibility, 156 blocks) and (c) (with visibility, 154 blocks) show the different images. (d) (w/o visibility, 37 blocks) and (e) (with visibility, 32blocks) show a zoom of the RMI data set with block boundaries drawn to illustrate different LODs.

TABLE III
THE STATISTICS OF SUMMARY TABLE SCHEMES.

	no encoding		**zigzag run-length encoding**	
data set	**space (overhead)**	**update time**	**space (overhead)**	**update time**
VisWoman	175MB (20.25%)	34s	9.22MB (1.07%)	5s
RMI	208MB (2.71%)	43s	44.1MB (0.57%)	13s



(a) (b) (c) (d)

Fig. 8. (a) (MSE-based, 107 blocks), (b) (SNRMSE-based, 108 blocks), and (c) (image-based, 106 blocks) show a zoom to the left foot, rendered close to full resolution. The reference image (d) is rendered with full resolution (151 blocks). White frames are drawn in (a) and (b) to indicate some of the differences.

(the blocks corresponding to the test bed have larger MSEs yet less visual importance than the blocks corresponding to the foot.). Although the SNRMSE-based one generates the result with the least noise, it leaves some portion rendered in lower resolution, which is discernable when compared with the reference image.

After applying the two strategies on visibility estimation for improving the performance (Section IV-E), we compared the timing of visibility estimation for CPU and GPU solutions. We evaluated the visibility of the multiresolution blocks in the data hierarchy for a wide variety of block budgets (from 10 to 11000), together with different combinations of translation, scale, and rotation. Table IV gives the upper bounds of the timing for the two data sets with four output image resolutions. The timing results show that the solution with the GPU is about three to four times faster than the CPU one.

B. Discussion

Compared with the traditional MSE-based and SNRMSE-based metric, our experience shows that for most of the cases, the image-based quality metric gives LODs better visual quality. This is especially true when the block budget is small (usually below 20%) compared with the number of blocks for full resolution. As one gradually increases the block budget, the three metrics generate closer results as expected. However, as shown in Fig. 8, we still get some improvement over the data-based metrics. Our image-based quality metric performs quite well even when the data contains noise. For example, the VisWoman data set contains noise from the 3D test bed. Fig. 4 shows that the image-based quality metric is insensitive to the noise and captures the structural distortion of the data, since more refinement effort was put on the blocks corresponding to the pelvis. This result is consistent with the image quality measure using structural similarity [30]. Finally, we compared our image-based quality metric with Guthe's screen-space metric (refer to Section II-B). We used the same estimated visibility with an implementation of the screen-space error and tested both data sets. The results in Fig. 9 show the advantage of our image-based quality metric over the screen-space error metric.

Our summary table scheme works well when the space overhead for storing the tables is small, compared with the

TABLE IV
THE TIMING OF VISIBILITY ESTIMATION WITH DIFFERENT OUTPUT IMAGE RESOLUTIONS.

	VisWoman						
image resolution	**CPU**	**draw**	**FB read**	**get avg**	**GPU**	**draw**	**SAT**
256×256	0.219s	8.72%	63.37%	27.91%	0.072s	9.72%	90.28%
512×512	0.578s	3.84%	58.59%	37.57%	0.151s	17.21%	82.79%
768×768	1.078s	2.56%	52.96%	44.48%	0.312s	17.63%	82.37%
1024×1024	1.531s	1.79%	42.83%	55.38%	0.487s	21.56%	78.44%
	RMI						
256×256	0.359s	6.41%	56.02%	37.57%	0.103s	9.71%	90.29%
512×512	0.828s	3.53%	52.16%	44.31%	0.185s	17.84%	82.16%
768×768	1.594s	2.09%	44.30%	53.61%	0.372s	16.13%	83.87%
1024×1024	2.338s	1.35%	34.47%	64.18%	0.538s	20.45%	79.55%
FB read = framebuffer read, **get avg** = get average occlusion							

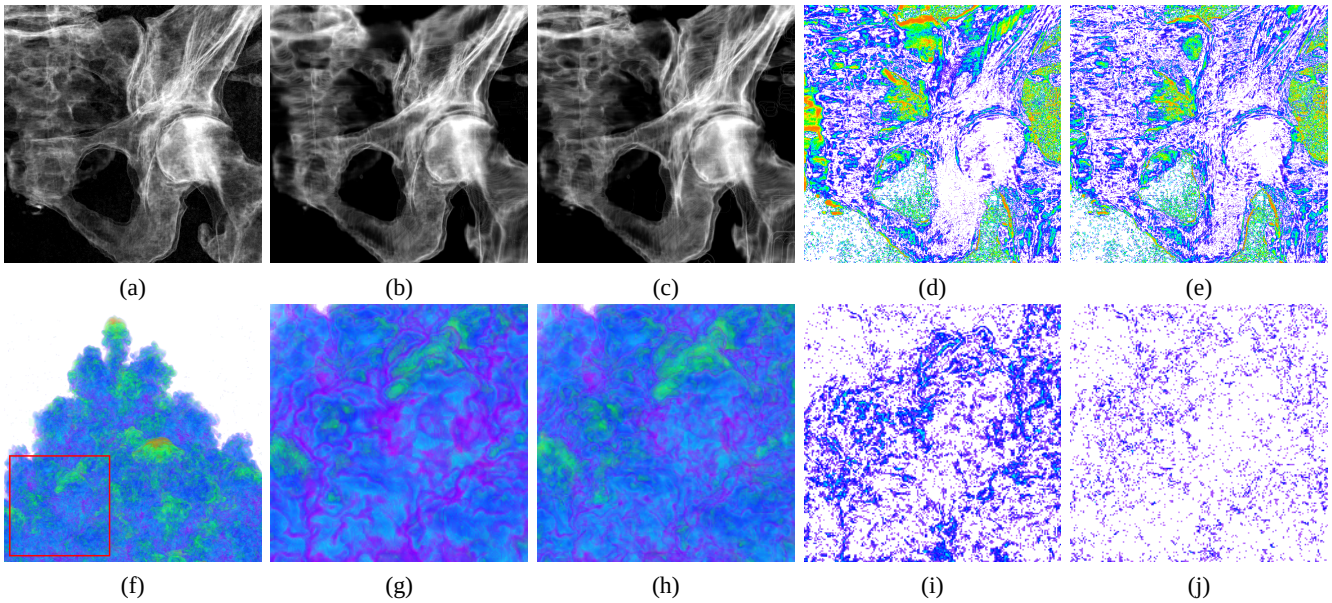


Fig. 9. First row: (a) (full resolution, 486 blocks), (b) (Guthe's screen-space metric, 56 blocks), and (c) (our image-based metric, 53 blocks) show a zoom to the left pelvis. (d) and (e) show the difference between (b) and (a), and (c) and (a) respectively. Second row: (f) shows a zoom rendered in high resolution (427 blocks). (g) (Guthe's screen-space metric, 43 blocks), and (h) (our image-based metric, 41 blocks) show a zoom of (f). (i) and (j) show the difference images for (g) and (h) respectively.

size of the input data set. In our experiments, good results were obtained with block sizes of 32^3 or 64^3 for a data set of size around 1024^3. This allows one to have histogram tables with sufficient entries (such as 256 in our experiments), while still keeping the scheme efficient. For a smaller block size, such as 16^3, we can reduce the number of entries in the histogram table, or use a simplified histogram to maintain an effective tradeoff between storage and processing requirements versus having enough precision for summary tables generation. On the other hand, our current solution is suitable for value-based transfer functions. There is a need of further research on generalizing the summary table scheme for multidimensional transfer functions.

To our knowledge, we are the first to utilize the GPU implementation of SATs for visibility estimation. We tested our GPU-based algorithm on the nVidia GeForce 7800 GT graphics card, which is based on the new generation PCI Express bus architecture. With PCI Express, the bandwidth between the CPU and the GPU increases to over 4GB per second in both upstream and downstream data transfers. In this case, framebuffer reads become less a constraint for the CPU-based solution. Still, it can be seen from Table IV that framebuffer reads take at least one third of the total time for the CPU solution. Our experiment reports that utilizing the GPU for visibility estimation, one can achieve a speedup up to four times.

For a typical output image resolution of 512^2, the summary of block classification is listed in Table V. For data-based metrics, the classification time is almost negligible since we only use the MSE or SNRMSE for block prioritization. For our image-based metric, taking into account the time for visibility estimation, we are able to prioritize all the multiresolution data blocks for LOD selection at a speed of 19.1kblocks/s for the VisWoman data set, and 14.0kblocks/s for the RMI data set. This result is comparable to the significance classification performance presented in [22], considering that we take a more

TABLE V
THE SUMMARY OF BLOCK CLASSIFICATION WITH 512^2 OUTPUT IMAGE RESOLUTION.

data set	VisWoman	RMI
block dimension	$32 \times 32 \times 64$	$128 \times 128 \times 64$
# blocks	9446	10499
data-based	0.028s	0.032s
visibility	0.151s	0.185s
prioritization	0.343s	0.563s
transfer function	5s	13s

refined and exact solution for block classification and visibility estimation. Note that the timing for visibility and prioritization gives the upper bound for all blocks. In actual rendering, the block budget is usually around tens to hundreds, and the classification time is much smaller than the upper bound (for instance, the prioritization time for the RMI data set is 0.016 second when the block budget is 1000). The classification of the blocks in the entire data hierarchy can be finished within seconds even if the user changes the transfer function at run time. This timing performance could be further improved with the support of transfer function preview at reduced resolutions. For example, the transfer function update time reduce to 1.7 seconds and 4.8 seconds from Table III for the VisWoman and RMI data sets respectively, if both use 64-level rather than 256-level transfer functions.

VI. CONCLUSION AND FUTURE WORK

The focus of this work is to develop an image-based LOD selection algorithm for large volumetric data, and produce images of better visual quality compared with traditional data-based LOD selection algorithms, under similar block budgets. In this paper, we have presented an interactive LOD selection and rendering algorithm through the use of an image-based quality metric. Experimental results on large scientific and medical data sets demonstrate the effectiveness and efficiency of our image-based LOD selection algorithm.

Our approach is promising due to its generality and flexibility. The summary table scheme greatly alleviates the dependence of the error calculation on the transfer function, and thus allows one to update the errors within seconds whenever the transfer function changes. The GPU reduction scheme for visibility estimation is not limited to multiresolution volume rendering, and is readily applicable to other large volume visualization scenarios that capitalize on the visibility information. Moreover, the hierarchical data representation and the user-specified budget for rendering make our LOD selection scheme suitable for time-critical multiresolution volume rendering and remote visualization applications. Finally, one can have different definitions and thus different ways of measurement for the multiresolution error in Eqn. 5, which we plan to explore more. In the future, we also would like to extend our method for large-scale time-varying data visualization.

APPENDIX
THE GPU IMPLEMENTATION OF SATS

Nowadays, GPUs and fragment programs support *render-to-texture* (RTT) with 32-bit floating-point channels for *pixel buffers* (pbuffers) as well as *framebuffer objects* (FBOs). This is important for the construction of SATs since the sums require more precision. Our implementation uses the `GL_EXT_framebuffer_object` extension to avoid the context switching of pbuffers. Given an input FBO, a SAT can be built by successively adding the columns from left to right and then the rows from bottom to top. However, this requires that the FBO is treated as both an input and an output texture, which highly depends on the kinds of hardware and graphics library available. To date, most implementations do not have this capability. Therefore, we take an alternative and build the SATs in passes with the support of FBOs having double auxiliary buffers: one is the input, and the other is the output. Both auxiliary buffers have the same size as the rendering framebuffer. At the beginning, the alpha values from the rendering buffer is mapped to the input buffer.

The construction of a SAT in the GPU is as follows: First of all, we operate on the columns. At the ith pass, each texel $T_i(x,y)$ in the output buffer is updated using two texels from the input buffer according to the following equation:

$$T_i(x,y) = T_{i-1}(x,y) + T_{i-1}(x - 2^{i-1}, y) \tag{11}$$

where sampling off texture returns zero and does not affect the sum. At the end of each pass, the auxiliary buffers are swapped. For a rendering screen with resolution of n^2, the number of passes needed is $\log_2(n)$. Then, the process is repeated over the rows in a similar way to complete the SAT construction.

The GPU implementation of SATs was first given by Green [11] from nVidia, where a simple scanline-based algorithm was presented and used for antialiasing in the traditional way. For an input table of size n^2, the number of passes is $2n$. Our implementation requires $2\log_2(n)$ passes with two sample reads per pass. Actually, this could be further improved by performing up to 16 sample reads per pass. Hensley et al. [14] performed study on the tradeoff between the number of rendering passes and the number of samples per pass. The optimal tradeoff between the number of passes and the cost per pass largely depends on the overhead of render target switches and the design of the texture cache on the target platform.

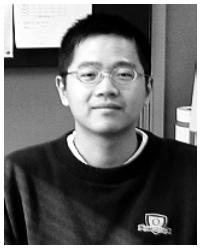
ACKNOWLEDGEMENTS

This work was supported by NSF ITR grant ACI-0325934, DOE Early Career Principal Investigator Award DE-FG02-03ER25572, NSF Career Award CCF-0346883, and Oak Ridge National Laboratory Contract 400045529. The VisWoman data set is courtesy of The National Library of Medicine, and the RMI data set is courtesy of Mark Duchaineau at Lawrence Livermore National Laboratory. The first author would like to thank Jian Huang, Ross J. Toedte, and Jinzhu Gao for initial discussions of this work. Finally, the authors would like to thank the reviewers for their helpful comments.

REFERENCES

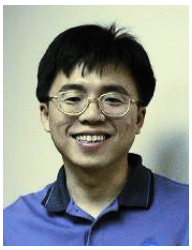
[1] I. Boada, I. Navazo, and R. Scopigno. Multiresolution Volume Visualization with a Texture-Based Octree. *The Visual Computer*, 17(3):185–197, 2001.

[2] D. Cohen-Or, Y. L. Chrysanthou, C. T. Silva, and F. Durand. A Survey of Visibility for Walkthrough Applications. *IEEE Trans. on Visualization & Computer Graphics*, 9(3):412–431, 2003.
[3] F. C. Crow. Summed-Area Tables for Texture Mapping. In *Proc. of ACM SIGGRAPH '84*, pages 207–212, 1984.
[4] J. El-Sana, N. Sokolovsky, and C. T. Silva. Integrating Occlusion Culling with View-Dependent Rendering. In *Proc. of IEEE Visualization '01*, pages 371–575, 2001.
[5] D. Ellsworth, L.-J. Chiang, and H.-W. Shen. Accelerating Time-Varying Hardware Volume Rendering Using TSP Trees and Color-Based Error Metrics. In *Proc. of IEEE Volume Visualization '00*, pages 119–129, 2000.
[6] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles & Practice in C (2nd Edition)*. Addison Wesley, 1995.
[7] A. Gaddipati, R. Machiraju, and R. Yagel. Steering Image Generation with Wavelet Based Perceptual Metric. In *Proc. of Eurographics '97*, 1997.
[8] J. Gao, J. Huang, H.-W. Shen, and J. A. Kohl. Visibility Culling Using Plenoptic Opacity Functions for Large Volume Visualization. In *Proc. of IEEE Visualization '03*, pages 341–348, 2003.
[9] M. H. Ghavamnia and X. D. Yang. Direct Rendering of Laplacian Pyramid Compressed Volume Data. In *Proc. of IEEE Visualization '95*, pages 192–199, 1995.
[10] A. S. Glassner. *Principle of Digital Image Synthesis, Volume 1*. Morgan Kaufmann, 1995.
[11] S. Green. Summed Area Tables using Graphics Hardware. In *Game Developers Conference '03*, 2003.
[12] S. Guthe and W. Straßer. Advanced Techniques for High-Quality Multi-Resolution Volume Rendering. *Computers & Graphics*, 28(1):51–58, 2004.
[13] S. Guthe, M. Wand, J. Gonser, and W. Straßer. Interactive Rendering of Large Volume Data Sets. In *Proc. of IEEE Visualization '02*, pages 53–60, 2002.
[14] J. Hensley, T. Scheuermann, G. Coombe, M. Singh, and A. Lastra. Fast Summed-Area Table Generation and its Applications. In *Proc. of Eurographics '05*, pages 547–555, 2005.
[15] C. E. Jacobs, A. Finkelstein, and D. H. Salesin. Fast Multiresolution Image Querying. In *Proc. of ACM SIGGRAPH '95*, pages 277–286, 1995.
[16] J. T. Klosowski and C. T. Silva. The Prioritized-Layered Projection Algorithm for Visible Set Estimation. *IEEE Trans. on Visualization & Computer Graphics*, 6(2):108–123, 2000.
[17] E. LaMar, B. Hamann, and K. I. Joy. Multiresolution Techniques for Interactive Texture-Based Volume Visualization. In *Proc. of IEEE Visualization '99*, pages 355–362, 1999.
[18] E. LaMar, B. Hamann, and K. I. Joy. Efficient Error Calculation for Multiresolution Texture-Based Volume Visualization. In *Hierarchical & Geometrical Methods in Scientific Visualization*, pages 51–62, 2003.
[19] D. Laur and P. Hanrahan. Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering. In *Proc. of ACM SIGGRAPH '91*, pages 285–288, 1991.
[20] M. Levoy. Efficient Ray Tracing of Volume Data. *ACM Trans. on Graphics*, 9(3):245–261, 1990.
[21] X. Li and H.-W. Shen. Time-Critical Multiresolution Volume Rendering Using 3D Texture Mapping Hardware. In *Proc. of IEEE Volume Visualization '02*, pages 29–36, 2002.
[22] P. Ljung, C. Lundström, A. Ynnerman, and K. Museth. Transfer Function Based Adaptive Decompression for Volume Rendering of Large Medical Data Sets. In *Proc. of IEEE Volume Visualization '04*, pages 25–32, 2004.
[23] N. Max. Optical Models for Direct Volume Rendering. *IEEE Trans. on Visualization & Computer Graphics*, 1(2):99–108, 1995.
[24] M. Meißner, J. Huang, D. Bartz, K. Muller, and R. Crawfis. A Practical Evaluation of Popular Volume Rendering Algorithms. In *Proc. of IEEE Volume Visualization '00*, pages 81–90, 2000.
[25] S. Muraki. Approximation and Rendering of Volume Data Using Wavelet Transforms. In *Proc. of IEEE Visualization '92*, pages 21–28, 1992.
[26] T. Porter and T. Duff. Compositing Digital Images. In *Proc. of ACM SIGGRAPH '84*, pages 253–259, 1984.
[27] N. Sahasrabudhe, J. E. West, R. Machiraju, and M. Janus. Structured Spatial Domain Image and Data Comparison Metrics. In *Proc. of IEEE Visualization '99*, pages 97–104, 1999.
[28] C. Ware. *Information Visualization: Perception for Design (2nd Edition)*. Morgan Kaufmann, 2004.
[29] C. Wang, J. Gao, and H.-W. Shen. Parallel Multiresolution Volume Rendering of Large Data Sets with Error-Guided Load Balancing. In *Proc. of Eurographics Parallel Graphics & Visualization '04*, pages 23–30, 2004.
[30] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Trans. on Image Processing*, 13(4):600–612, 2004.
[31] R. Westermann. A Multiresolution Framework for Volume Rendering. In *Proc. of IEEE Volume Visualization '94*, pages 51–58, 1994.
[32] J. Wilhelms and A. van Gelder. Multi-Dimensional Trees for Controlled Volume Rendering and Compression. In *Proc. of IEEE Volume Visualization '94*, pages 27–34, 1994.
[33] H. Zhou, M. Chen, and M. F. Webster. Comparative Evaluation of Visualization and Experimental Results Using Image Comparison Metrics. In *Proc. of IEEE Visualization '02*, pages 315–322, 2002.



Chaoli Wang received the BE and ME degrees in computer science from Fuzhou University, China in 1998 and 2001 respectively. Currently, he is a PhD candidate in the Department of Computer Science and Engineering at The Ohio State University. His research interests are computer graphics and visualization, with focus on developing algorithms for managing and rendering large-scale three-dimensional and time-varying data. He is a student member of the IEEE.

Antonio Garcia received the BS degree in systems engineering from Universidad Particular "Antenor Orrego" in 1993, the MS degree in computer science from The Ohio State University in 1999. He completed his PhD dissertation at The Ohio State University in 2005. His research was focused on parallel rendering, volume rendering, and GPU programming. Currently, he is working for Intel Corporation in the production, debugging and testing of display drivers for Intel's chipsets and GPUs.



Han-Wei Shen received his BS degree from Department of Computer Science and Information Engineering at National Taiwan University in 1988, the MS degree in computer science from the State University of New York at Stony Brook in 1992, and the PhD degree in computer science from the University of Utah in 1998. From 1996 to 1999, he was a research scientist at NASA Ames Research Center in Mountain View California. He is currently an Associate Professor at The Ohio State University. His primary research interests are scientific visualization and computer graphics. Professor Shen is a winner of US Department of Energy's Early Career Principal Investigator Award and National Science Foundation's CAREER award. He also won an Outstanding Teaching award in the Department of Computer Science and Engineering at The Ohio State University.